Implementing the BeanJuice Service

The simple BeanJuice Service takes a single parameter (your name) and returns a message with your name appended.

The Build process requires the two XML build files shown in the illustration above.

The \qube\service\build.xml file

```
  …
  <filelist id="sub.projects" dir="${basedir}">
    …
    <file name="example-service-beanjuice"/>
    …
  </filelist>
</project>
```

The \qube\service\example-service-beanjuice\build.xml file

The \qube\service\example-service-beanjuice\build.xml file must be customised for each Service. The following extract shows the two places where this file is customised for this Service.

```
<project name="beanjuice-service" default="build" basedir=".">
  …
  <property name="name" value="service-beanjuice"/>
  …
</project>
```

The BeanJuiceService.java source file

The following extract is the BeanJuiceService.java source file that defines the Interface for the BeanJuice Service.

```
/**
  * @(#)BeanJuiceService.java
  *
  * Copyright (c)2007 Quative Ltd. All Rights Reserved.
  */
package tv.quative.service.beanjuice.api;
import tv.quative.service.Service;
  /**
  * An example of a simple Service.
```

```
  * @author Joe Dorward
  */
public interface BeanJuiceService extends Service {
  /**
  * Returns the 'Bean Juice' message.
  * @param name - the name of the User
  * @return String - the 'Bean Juice' message and the User name
  * @throws Exception - on error
  */
  String beanJuiceMessage(String name) throws Exception;
}
```

The BeanJuiceServiceImpl.java source file

The following extract is the BeanJuiceServiceImpl.java source file that defines the Implementation of the BeanJuice Service.

```
/**
  * @(#)BeanJuiceServiceImpl.java
  *
  * Copyright (c)2007 Quative Ltd. All Rights Reserved.
  */
package tv.quative.service.beanjuice.impl;
import tv.quative.service.BaseService;
import tv.quative.service.QubeMethod;
import tv.quative.service.QubeService;
import tv.quative.service.beanjuice.api.BeanJuiceService;
  /**
  * An example of a simple Service.
  * @author Joe Dorward
  * @version 1.0
  */
@QubeService(name = "beanJuiceService", ws = true, hue = true)
public class BeanJuiceServiceImpl extends BaseService implements BeanJuiceService {
  /**
  * Returns the 'Bean Juice' message.
  * @param name - the name of the User
  * @return String - the 'Bean Juice' message and the User name
  * @throws Exception - on error
```

```
  */
  @QubeMethod (ws = true, hue = true, roles = "all")
  public String beanJuiceMessage(final String name) throws Exception {
    return "Bring me the Bean Juice! " + name;
  }
}
```

By convention the name of a Class implemented as a Service is named following the form:

<service-name>ServiceImpl

In the above extract (the @QubeService annotation):

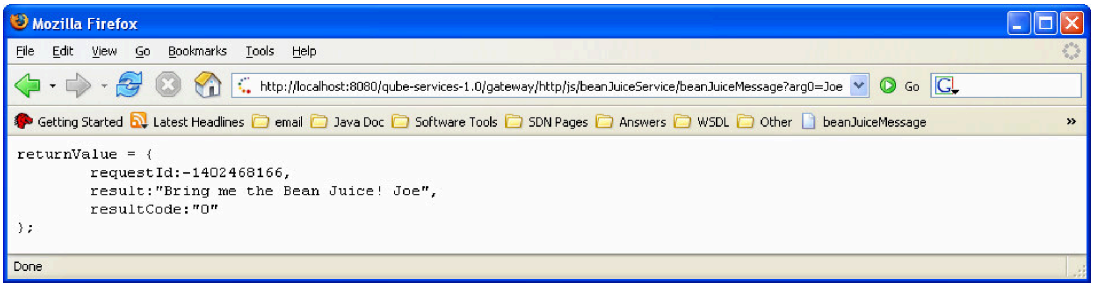
- The @QubeService annotation designates the Class as a Service
- The string value beanJuiceService of the name attribute is the name of the Service
- The string value true of the ws attribute means the Service will be accessible using Web Services – false would make it inaccessible
- The string value true of the hue attribute means the Service will be accessible using HTTP – false would make it inaccessible

Building and deploying the Service

Although building and deploying Services is beyond the scope of this document – the next step assumes you've done that.

Calling the BeanJuice Service over HTTP

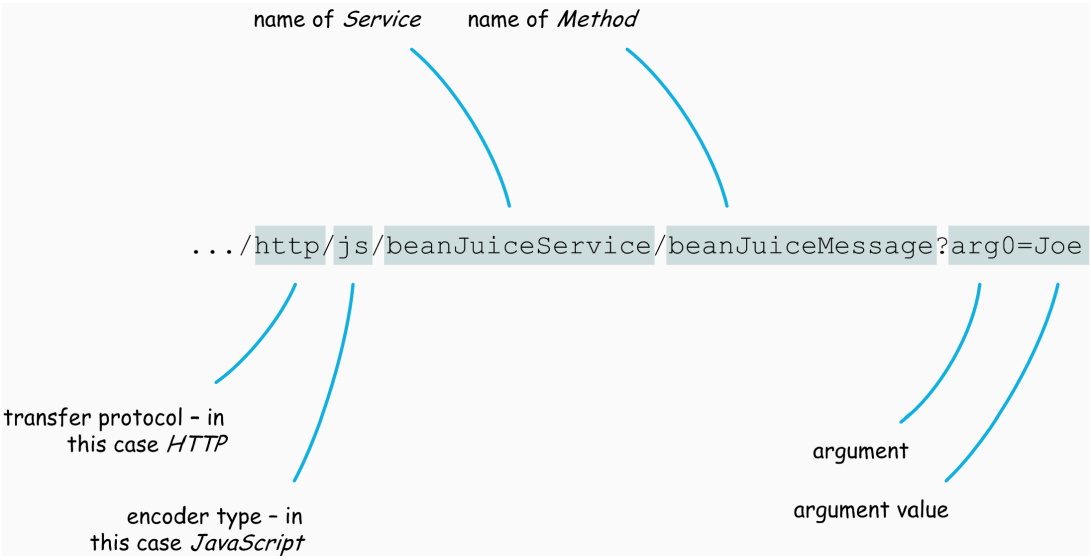
Because the BeanJuice Service is exposed to the HUE Gateway Interface – you can test it using a standard web browser by typing a URL into its Address Box. The following illustration shows how to call the BeanJuice Service this way.



Calling the BeanJuice Service

The above extract shows the URL used to call the BeanJuice Service and the result of that call – a JavaScript object – in the web browser.

The following illustration explains the component pieces of the URL shown above.



Calling the BeanJuice Service (URL Callout)